Scala Quick Intro For Chiselers

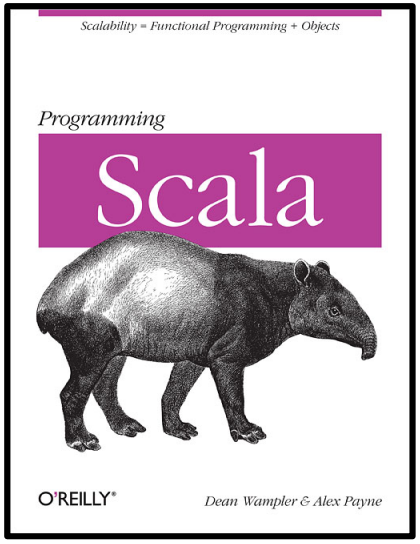
Jonathan Bachrach

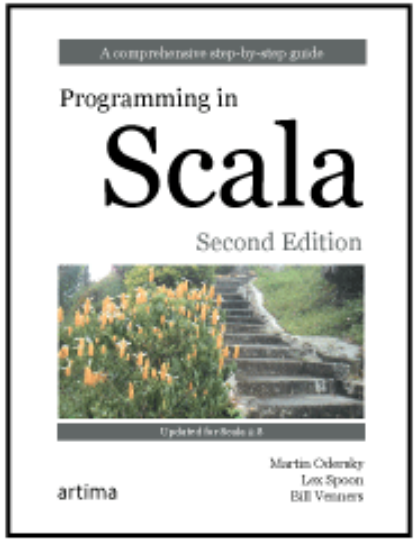
EECS UC Berkeley

August 29, 2014

The Scala Programming Language

- Object Oriented
  - Factory Objects, Classes
  - Traits, overloading etc
  - Strongly typed with type inference
- Functional
  - Higher order functions
  - Anonymous functions
  - Currying etc
- Extensible
  - Domain Specific Languages (DSLs)
- Compiled to JVM
  - Good performance
  - Great Java interoperability
  - Mature debugging, execution environments
- Growing Popularity
  - Twitter
  - many Universities





Scala Bindings


```
// constant
val x = 1
val (x, y) = (1, 2)

// variable
var y = 2
y = 3
```

```
// Array's
val tbl = new Array[Int](256)
tbl(0) = 32
val y = tbl(0)
val n = tbl.length

// ArrayBuffer's
import scala.collection.mutable.ArrayBuffer
val buf = new ArrayBuffer[Int]()
buf += 12
val z = buf(0)
val l = buf.length

// List's
val els = List(1, 2, 3)
val els2 = x :: y :: y :: Nil
val a :: b :: c :: Nil = els
val m = els.length

// Tuple's
val (x, y, z) = (1, 2, 3)
```

Scala Maps and Sets


```
import scala.collection.mutable.HashMap

val vars = new HashMap[String, Int]()
vars("a") = 1
vars("b") = 2
vars.size
vars.contains("c")
vars.getOrElse("c", -1)
vars.keys
vars.values
```

```
import scala.collection.mutable.HashSet

val keys = new HashSet[Int]()
keys += 1
keys += 5
keys.size -> 2
keys.contains(2) -> false
```

Scala Iteration


```
val tbl = new Array[Int](256)

// loop over all indices
for (i <- 0 until tbl.length)
  tbl(i) = i

// loop of each sequence element
val tbl2 = new ArrayBuffer[Int]
for (e <- tbl)
  tbl2 += 2*e

// loop over hashmap key / values
for ((x, y) <- vars)
  println("K " + x + " V " + y)
```

Scala Functions


```
// simple scaling function, e.g., x2(3) => 6
def x2 (x: Int) = 2 * x
```

```
// more complicated function with statements
def f (x: Int, y: Int) = {
  val xy = x + y;
  if (x < y) xy else -xy
}
```

```
// simple scaling function, e.g., x2(3) => 6
def x2 (x: Int) = 2 * x
```

```
// produce list of 2 * elements, e.g., x2list(List(1, 2, 3)) => List(2, 4, 6)
def x2list (xs: List[Int]) = xs.map(x2)
```

```
// simple addition function, e.g., add(1, 2) => 3
def add (x: Int, y: Int) = x + y
```

```
// sum all elements using pairwise reduction, e.g., sum(List(1, 2, 3)) => 6
def sum (xs: List[Int]) = xs.foldLeft(0)(add)
```

```
class Blimp(r: Double) {
  val rad = r
  println("Another Blimp")
}

new Blimp(10.0)
```

```
class Zep(h: Boolean, r: Double) extends Blimp(r) {
  val isHydrogen = h
}

new Zep(true, 100.0)
```

Scala Singleton Objects

- like Java class methods
- for top level methods

```
object Blimp {
  var numBlimps = 0
  def apply(r: Double) = {
    numBlimps += 1
    new Blimp(r)
  }
}

Blimp.numBlimps
Blimp(10.0)
```